**Tema:**

***Tabele in iskalni algoritmi***

**Vsebine:**

*Linearno iskanje v neurejeni tabeli*

*Binarno iskanje*

*Linijska (serijska) vizualizacija*

**Naloga 1**

Dana je razredna metoda s kodnim/razvojnim imenom x\_32\_v4.

static int x\_32\_v4(long abc){

int ll=1;

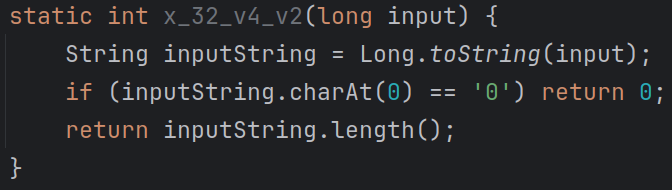
for(;(abc/=10)!=0;ll++);

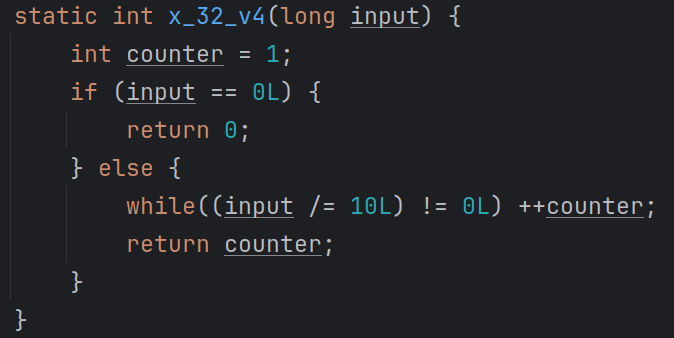
return Math.max(1,ll);

}

* Analizirajte njeno delovanje in ugotovite, kaj počne  
  **Metoda vzame eno število tipa long in jo deli z 10 dokler ne dobi rezultata 0, za vsako delitev poveča spremenljivko ll za 1 in nakoncu vrne to število. Metoda ugotovi število števk v danem long številu.**
* Glede na ugotovitev dokažite pravilnost (ugotovljenega) delovanja (dokazi bazirajo na osnovi sledi izvajanj).Dokazi morajo pokrivati/dokazovati pravilnost/ vseh vstopnih možnosti za dani postopek (oziroma vsako vrednost argumenta).

**Program izpiše število števk za 1 premalo, to se zgodi zaradi zaokroževanja pri deljenju, ko mora java zaokrožiti števila jih vedno zaokroži navzdol, zaren če tega posebej ne spremenimo.**

* Spišite demonstracijski program, ki bo pokazal pričakovano delovanje metode pri klicu s poljubno celoštevilsko vrednostjo.

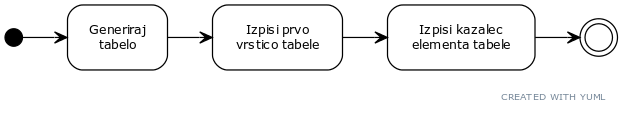
Ali:

* Obrazložite, zakaj iz metode vračamo večjo od specificiranih vrednosti.  
  **?**
* Predlagajte ime metode, ki bo dovolj opisno za funkcijo, ki jo opravlja.

**steviloStevk**

**Naloga 2**

Dan javanski program izvaja sekvenco aktivnosti, kot je prikazano na diagramu:

Dopolnite dano kodo z realizacijo ne-implementiranih metod. Pri tem predpostavite:

public class Vizualizacija00 {

static int dolzinaTabele=12

public static void main(String[] args){

int[] tab=generirajTabelo(dolzinaTabele);

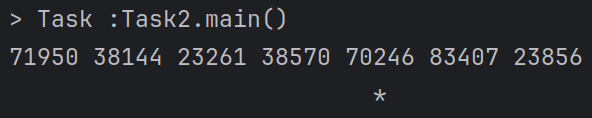
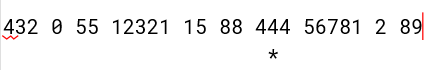
izpisiPrvoVrsticoTabele(tab);

izpisiKazelecElemTabele(tab,6);

}

}

* tabela lahko vsebuje zgolj eno do 5 mestna pseudo-naključna pozitivna števila z vključeno vrednostjo 0;
* izpis prve vrstice tabele izpiše števila v eni sami vrstici, pri tem se med posameznimi števili izpiše en sam presledek
* izpis kazalca se izvrši pod izpisom tabele na sredini pod število, na katerega kaže. Npr.

kaže na element z zaporedno številko (indeksom) 6, kot je dano s primerom delne realizacije programa, danega v nadaljevanju:

**Naloga 3**

Med obe izpisni vrstici v razredu Vizualizacija00 vrinite naslednje zaporedje stavkov:

try{

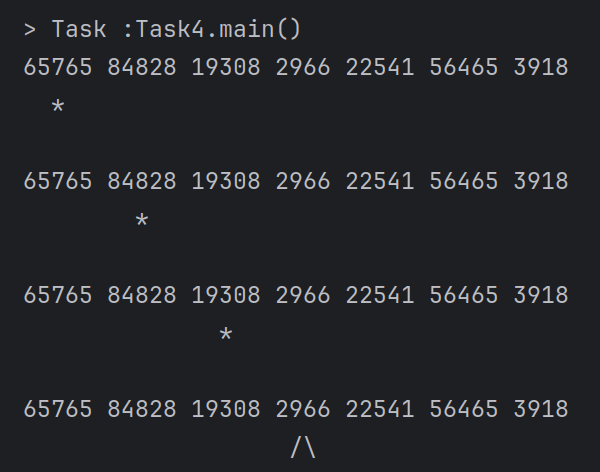
Thread.sleep(2000);

} catch(InterruptedException e){}

Obrazložite, kaj se je med izvajanjem zgodilo. Povečajte še vrednost argumenta metode sleep na npr. 7000 in opazujte razliko.  
**Metoda Thread.sleep(2000) program zamrzne za 2 sekundi, z 7000 bi to bilo 7s.**

**Naloga 4**

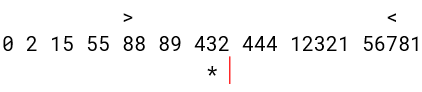
Z zaporednimi klici izpisovalnih metod iz glavne metode razreda Vizualizacija00 je mogoče izvesti vizualizacijo postopka zaporednega iskanja, korak po korak. Izvedite jo. Pri tem podajte primer za iskanja za obstoječo vrednost in za neobstoječo vrednost. Naj bo zakasnitev med posameznimi koraki izvajanja vsaj 1.5s.

****

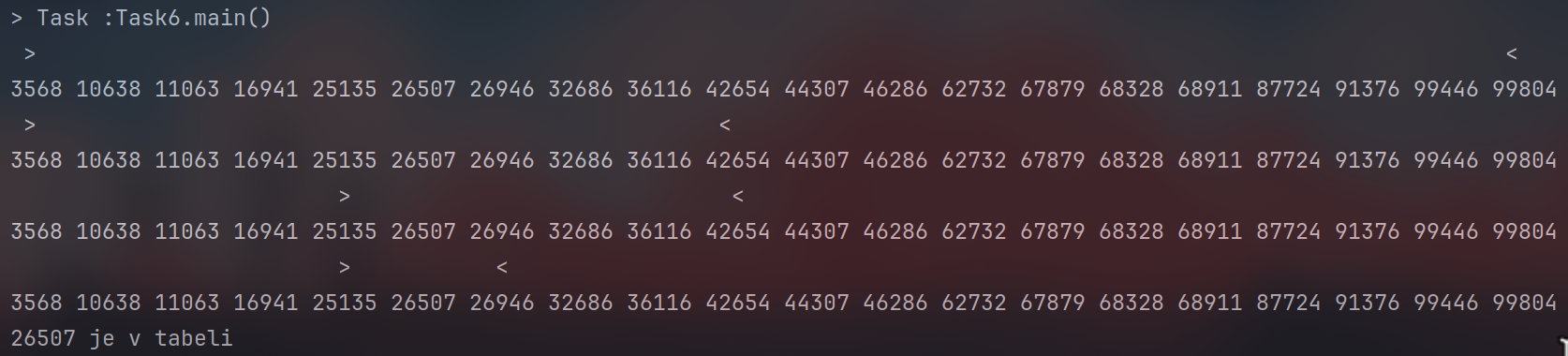
**Naloga 5**

Na podoben način, kot je izvedena animacija vizualizacije postopka v nalogi 4, izvedite tudi za binarno iskanje. Za to:

* Dodajte metodo generirajUrejenoTabelo/1
* Dodajte kot prvi klic izpisnih metod še klic metode izpisiPredPrvoVrsticoTabele(tab,spod,zgor); , ki bo izrisal meje iskalnega zaporedja v postopku binarnega iskanja v vrstici nad vsebino elementov tabele. Npr.:



Razmislite tudi o spremembi znaka, ko sta obe meji postavljeni na isti element tabele.



**Naloga 6**

Poglejmo si, kako na zaporedju iz naloge 5 deluje metoda binarySearch razreda priročnih metod za delo s tabelami [java.util.Arrays](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Arrays.html)  :

* Spišite enostaven program, ki bo z omenjeno funkcijo v omenjenen zaporedju iskal vrednosti 88, 0, 56781 in nazadnje 5555 (opazite lahko, da so vrednosti testni primeri ). Ugotovite, kaj metoda vrača v posameznih primerih iskanja. **Metoda vrača indekse testnih primerov, če pa testnega primera ni vrne negativno število kje približno bi se moral nahajati.**

